Eötvös Loránd University Faculty of Informatics Department of Numerical Analysis



Using IoT Technology to Simulate Smart House Environment

*Supervisor:*

ILLÉS ZOLTÁN

Doctor/ Lecturer

*Author:*

LE THANH DUC

Computer Science BSc

*Budapest, 2021*

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

Thesis Registration Form

Student's Data:

**Student's Name:** Le Thanh Duc

**Student's Neptun code:** YZ49Y8

Course Data:

**Student's Major:** Computer Science BSc

I have an internal supervisor

***Internal Supervisor's Name:*** *ILLÉS ZOLTÁN*

*Supervisor's Home Institution: ELTE University*

*Address of Supervisor's Home Institution: Room: 2.403 - 1117 Budapest, Pázmány Péter sétány 1 / c.*

*Supervisor's Position and Degree: Doctor /Lecturer*

**Thesis Title:** Using IoT Technology to simulate SmartHouse Environment

Topic of the Thesis:

*(Upon consulting with your supervisor, give a 150-300-word-long synopsis os your planned thesis. )*

Based on what Professor/Dr. Zoltán and I discussed through email plus the topic was also inspired to me by my father, I am going to simulate the smart house environment using IoT Technology and it will be related to AWS Cloud or any other cloud services. I plan to use Raspberry Pi 3 as a local server containing OpenHab2 which uses MQTT to communicate with for example SonOff switch and other sensor devices within the house. In order to simulate the communication with other devices within the house, I will also code NodeMCU Esp8266 using Arduino language to control some lights or small fans/ motors. In term of cloud, I am thinking about using Amazon Alexa or Google Home and its Cloud Service Provider to control the house by voice. All of the devices will be compacted and simulated by a big box to be presented as a small smart house. There will be a dynamic website to control it via internet from distances. So far, this is what I have planned to do. Hopefully, this would be an interesting topic and a successful thesis.

Budapest, 2020.10.17.

Contents

[Chapter 1 3](#_Toc70635039)

[Introduction 3](#_Toc70635040)

[1.1. Problem statement 3](#_Toc70635041)

[1.2. Aim specification 4](#_Toc70635042)

[Chapter 2 4](#_Toc70635043)

[Bibliographical research 5](#_Toc70635044)

[2.1. Gateway and protocols 5](#_Toc70635045)

[2.1.1. HTTP 5](#_Toc70635046)

[2.1.2. MQTT 7](#_Toc70635047)

[2.2. Devices 8](#_Toc70635048)

[2.2.1. Raspberry Pi 8](#_Toc70635049)

[2.2.2. Node MCU - ESP8266 9](#_Toc70635050)

[2.2.3. ESP32 CAMERA 12](#_Toc70635051)

[2.2.4. Sonoff 4CH Switches 13](#_Toc70635052)

[2.2.5. HR202 Sensor 14](#_Toc70635053)

[2.2.6. HC- SR505 Sensor 15](#_Toc70635054)

[2.2.7. Diffused RGB LED 16](#_Toc70635055)

[2.2.8. Google Home Mini 16](#_Toc70635056)

[2.3. Home automation platforms 16](#_Toc70635057)

[2.3.1. openHAB 16](#_Toc70635058)

[2.3.2. myopenHAB 17](#_Toc70635059)

[2.4. System plan 18](#_Toc70635060)

[2.5. Decision- making process 19](#_Toc70635061)

[2.6. User Documentation 20](#_Toc70635062)

[Chapter 3 23](#_Toc70635063)

[Developer Documentation 23](#_Toc70635064)

[3.1. Installation 23](#_Toc70635065)

[3.1.1. Raspberry Pi 3 openHABian setup 23](#_Toc70635066)

[3.1.2. Raspberry Pi 3 Apache2 setup 24](#_Toc70635067)

[Bibliography 25](#_Toc70635068)

# Chapter 1

# Introduction

## Problem statement

The following work goes into detail on how Internet of Things technologies work and can aid in building smart house environment while minimizing the amount of electricity and resources used in order to give user the best experience in their own house.

The bibliography covers the state of art and available technologies such as communication protocols and their characteristics, as well as commercially available Internet of Things devices and sensors as potentially viable candidates in helping to achieve the goal of this work.

With the knowledge gathered from many researches and testing, an educated decision was made on how to develop the project. The development plan goes into detail how an idea solution fitting the requirements would work for the given use case and outlines the functions of the system by specifying the system’s components and its main deployable elements. Additionally, the planning phase sets the guidelines for the user interface design and interaction, finishing with a sequential estimation of the development efforts needed to complete this project.

The development section shelters the main parts of understanding the created strategies and plans. Beginning with necessary steps for putting in place the primary infrastructure required to make the proof of concept work. After that, it goes into detail through the required configuration of the physical devices existing in the system, followed by the development of the web application which is used for user interface and user interacting with the system.

The works presented in this thesis are a part of my own project with deployed physical system in a realistic house. The system mainly demonstrates how Internet of Things solution works and could provide a way of controlling smart house’s environments also its behavior to make life more comfortable and easier with as little manual intervention as possible.

# Aim specification

There are many great opportunities that are available with the Internet of Things. The key promise of IoT is quick and continuous access to almost anywhere, which means we did not find valid in our research. It's all you need is a basic internet connection and a simple piece of hardware to do this. Let's say, for example, users could use Google Maps to figure out where they are instead of asking anyone on the street, make a reservation, or consulting our calendars automation is conveniently and inexpensively built into the Internet of Things. as a result, smart offices decrease the number of routine activities for the workers, and free them up their time and effort to engage in other tasks that are more demanding. Also in another point of view, it is better for time management, the Internet of Things is a fantastic time-saving device. It might be engaged in a conversation or have on the phone or reading our favorite news while we're in the car, we can also look for something in the online shops while out and websites, and place an order, and call it if we like. People will end up with plenty of time on our hands in the long run.

<https://readwrite.com/2020/06/24/the-biggest-advantages-and-disadvantages-of-iot/>

By looking at various contexts in which things like the business world, the health care industry, and even people's own particular emotional lives, this study seeks to discover and extend principles of which can help people become universal practices. This is a simple illustration of how things like the Internet of Things function in the real world.

The aim of this project is to use Internet of Things (IoT) to make life at home smarter. This sensor is designed to gather information about the environment around the building, such as temperature and humidity, and note if there's movement, as well as any changes in it. For instance, if you want a particular theme to be automatically applied at a particular time of day or night, this will also provide you with the ability to allow that. It is probable that an extension of this strategy or anything similar might lead to an unusually abundant supply of resources in the local region.

# Chapter 2

# Research Methods

## 2.1. Gateway and protocols

### 2.1.1. HTTP

Hypertext Transfer Protocol – the protocol for the World Wide Web which is proposed by Tim Berners- Lee. It was a global information network allowing the flow of websites, and files to occur in a consistent manner through many languages. The W3C has maintained it ever since.

HTTP works by having servers and user agents that connect to them, enabling the agents to consume the data. Server location is finished by using a Uniform Resource Identifier (URI) or Uniform Resource Locator (URL) which are subset of the former set.

[1] Y. Lafon, "HTTP - Hypertext Transfer Protocol," 2003. [Online]. Available: https://www.w3.org/Protocols/. [Accessed 13 December 2019].

[2] R. Fielding, "Hypertext Transfer Protocol -- HTTP/1.1," June 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616. [Accessed 13 December 2019].

The Hypertext Transfer Protocol follows the request-response model where a device, such as a web browser or app, sends a request to the HTTP server. either the response includes a status code stating whether the request was successful, or there was an internal error in the server, or not; in the latter case, the requested item might not be available.

HTTP specifies a number of methods for requesting server query, creating, modified, deleting or providing meaningful necessary data and accessibility for the server.

These methods are:

* GET: Requests the required resource representation. These applications should only get data and have no other impact.
* HEAD: However, the response body is not returned, similar to GET. Used to search for metadata and return a smaller payload size.
* POST: Adds to the server a new resource, as defined in the application body.
* PUT: Used in order to add a given URI resource. A new resource is provided if the URI is not available. The resource is changed if the URI is present.
* PATCH: used to implement partial resource modifications.
* DELETE: Deletes the resource listed.
* TRACE: Used for debugging via loopback testing of the message along the path towards the target resource.
* OPTIONS: Used on the specified URI to query the list of HTTP support methods. This is used to implement security restrictions on some resources by Cross-Origin Resource Sharing (CORS). The OPTIONS question is referred to as a pre-flight request here.
* CONNECT: Creates a TCP/IP tunnel, used to facilitate SSL-encrypted communication through an unencrypted HTTP proxy.

To ensure end-to-end security and to guard against man-in-the-middle attacks, a standard HTTP communication is developed over Secure Sockets Layer (SSL), or the more commonly used Transport Layer Security (TLS) (TLS). This enables the entire HTTP protocol to be encrypted and transmitted over SSL or TLS. These encryption schemes normally use long-term private and public keys to generate a short-term session key, which is then used to encrypt communication between the server and the client.

HTTPS is critical in publicly accessible networks, such as Wi-Fi hotspots found in cafes, libraries, and universities, since anyone linked to such a network can reasonably easily eavesdrop on communication transmitted over simple unencrypted HTTP. The same logic applies to situations in which Wireless Local Area Networks (WLANs) are needed to allow access to a large number of devices in an area that may be frequented by untrusted personnel.

Within the scope of this work, the HTTP(S) protocol can be used to create contact between components, most notably between a web-based application and a backend service that exposes application programming interfaces (APIs) through this protocol, as it is the de-facto requirement for web-based services at the time of writing this study.

The Remote Procedure Call (RPC) protocol is an alternative to HTTP. However, due to the RFC protocol's age and the greater prevalence of HTTP, HTTP support in modern programming languages is superior to RFC support.

### 2.1.2. MQTT

MQTT- Message Queueing Telemetry Transport, is a publish-subscribe messaging protocol. It was developed as a lightweight messaging transport protocol atop the TCP/IP protocol suite. It is intended for use in situations involving machine-to-machine (M2M) communication and the Internet of Things. MQTT provides connections for remote areas that need a small code footprint or have very limited network bandwidth.

Clients interact with a server, which is sometimes referred to as a "broker," in a MQTT scheme. Any client may connect to the broker and either publish or subscribe to topics. Topics are classes of information that are coordinated and shared. A client subscribes to a subject, and when the broker receives a message published by a client within that topic, it distributes the information contained in the message to all topic subscribers. The publisher does not need to know anything about the subscribers, and the subscribers do not need to know anything about the publishers. Messages in a subject with no subscribers are discarded by the broker.

MQTT's data transmission is based on the TCP protocol. MQTT-SN is a version of MQTT that is used over insecure transport protocols like UDP or Bluetooth. MQTT does not have any encryption or authentication mechanisms and sends out link credentials in plain text format. The TCP protocol, which sits underneath MQTT, often provides these steps.

The ISO/IEC PRF 20922 specification defines this protocol. The Organization for the Advancement of Structured Information Standards (OASIS) is in charge of defining new standards enhancements, and it recently released the official MQTT v5.0 specification, which represents a major step forward in the MQTT protocol's refinement. OASIS took special care to maintain backward compatibility when updating the standard to include features such as Message Properties, which adds metadata to the header of a MQTT message, Shared Subscriptions, which allows for load balancing across multiple clients, Message and Session Expiry, Topic Aliases, which provides mappings between topic strings in messages, and Message and Session Expiry.

Because of its low power consumption and small data packet size, MQTT is used by many IoT solutions to create communications between devices.

In addition, unlike HTTP, a MQTT link does not close after each request, leaving the communication channel open and incurring the overhead of opening and closing connections.

The MQTT protocol is simple to set up, and there are a variety of MQTT brokers on the market, making it an appealing candidate for choosing a protocol over which some of the IoT devices mentioned by this project could communicate. However, since the MQTT protocol is based on TCP, it necessitates several extra steps before a link capable of exchanging data can be created, which can affect “wake-up” and communication times. This may also have an effect on the device's power consumption, as many IoT devices have minimal processing power, making the process of creating a new TCP session more resource intensive and, as a result, increasing the device's total energy consumption.

## 2.2. Devices

### 2.2.1. Raspberry Pi

It serves as, in a sense, as a miniature, and effectively, a functioning machine intended for use in schools and developing countries, and was created by the Raspberry Pi Foundation with the aim of educating the next generation of computer science engineers and programmers about computing fundamentals. Moreover, the flexibility and practicality of the product surpassed the manufacturer's expectations in terms of sales and use cases.

By the time you are reading this, 4 big generations of Raspberry Pi have already been marketed to the public. With a Broadcom System-on-Chip (SoC) on which is an integrated Graphical Unit (GPU) to bring benefits to the table, all SoCs are assembled with a combination of a Broadcom SoCs and CPUs that provide the graphics processing benefit (GPU). There are from one to four USB ports on each board. This includes two HDMI and one composite video ports as well as a traditional 1/2.5" headphone and microphone port. Later versions of the Raspberry Pi added General Purpose Input/Output (GPIO) pins to make connecting and writing to additional devices easier and allow for greater programming flexibility. Later on, in the product development phase, B and B+ model versions were equipped with Ethernet networking capabilities. Since the Raspberry Pi 3 and Pi Zero have Wi-Fi access, it is also possible to add it to other versions. There are a variety of peripherals available for the Raspberry Pi, including cameras, screens, and other boards, it is not necessary to get any of these individually.

Moreover, Bluetooth and Bluetooth Low Energy communication are also supported by more recent Raspberry Pi devices. Some communications can be established by using add-in boards, commonly known as dongles, for the Raspberry Pi, or by attaching one of the available ports on the computer. Raspberry has powerful processing chips, which is a bonus for MQTT because of its minimal involvement on the performance of devices in general, allows single devices to handle multiple MQTT connections without requiring major disruption to their operations.

The Raspberry Pi Foundation provides Raspbian, a Debian-based Linux distribution, as the default operating system for these computers. The programming languages Python and Scratch are featured as the primary and default programming languages on this distribution due to their ease of learning, with support for several other languages provided by the operating system. The Raspberry Pi can also run a variety of other operating systems, including Windows 10 IoT Core, FreeBSD, NetBSD, RISC OS, Android Stuff, SUSE, Fedora, Xubuntu, and many others.

The Raspberry Pi community is a fundamental component of the ecosystem as a whole. It is considered very rich and proactive because it is a relatively large group of developers, instructors, users, and enthusiasts who contribute to open source projects and maintain them, allowing companies and private individuals to actively invent and build new projects either for or with the help of the community.

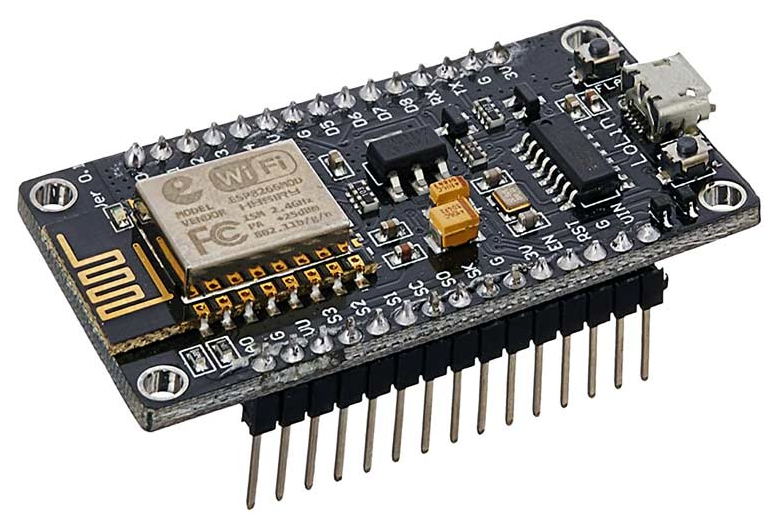
Additionally, because of the Raspberry Pi devices' low price, they have become very popular among home automation enthusiasts. People who are interested in technology are changing and expanding the capabilities of Raspberry Pi devices in order to create more versatile, expandable, and cost-effective home automation solutions than those available on the commercial market.

Indeed, “ModBerry” is a computer based on the Raspberry Pi that was launched by TECHBASE in 2014, includes serial ports, CAN and Wire-1 buses, as well as digital and analog I/O, both of which are commonly used in the automation industry. This device's architecture makes it suitable for use in harsh industrial conditions, addressing the Raspberry Pi's shortcomings in terms of industrial IoT solutions.

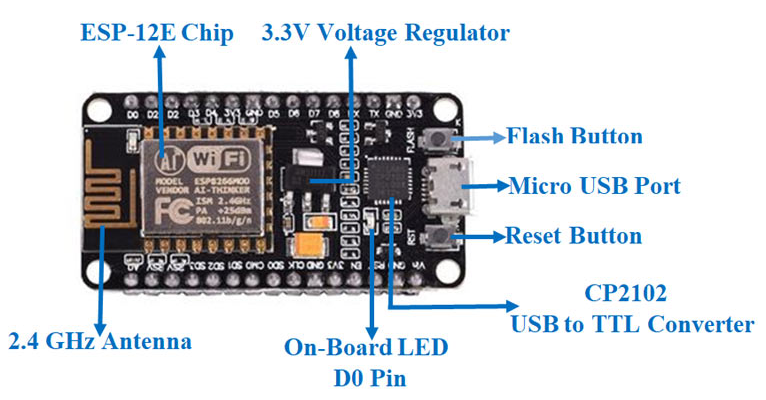
The Raspberry Pi is considered to be potentially one of the most useful devices when bringing the concept behind this project to existence due to the excellent group driving the ecosystem, sponsorship, open-source software, and availability of Raspberry Pi devices, coupled with their affordability and ease of functional extension.

### 2.2.2. Node MCU - ESP8266

NodeMCU is an open-source Lua-based firmware and development board designed specifically for Internet of Things (IoT) applications. It includes firmware that runs on Espressif Systems' ESP8266 Wi-Fi SoC and hardware based on the ESP-12 board.



The NodeMCU ESP8266 development board comes with a specific ESP-12 module which includes Tensilica 32-All RISC chip with Tensil-micro XTSU microprocessor (name of one of the supplier's chips). This microprocessor runs on an adjustable clock frequency of 80MHz to 160MHz and supports RTOS. To store data and programs, the NodeMCU has 128 KB of RAM and 4MB of Flash memory. It is suitable for IoT projects due to its high processing speed, built-in Wi-Fi / Bluetooth, and Deep Sleep Operating features. The NodeMCU is operated by a Micro USB jack and a VIN pin (External Supply Pin). It has interfaces for UART, SPI, and I2C.



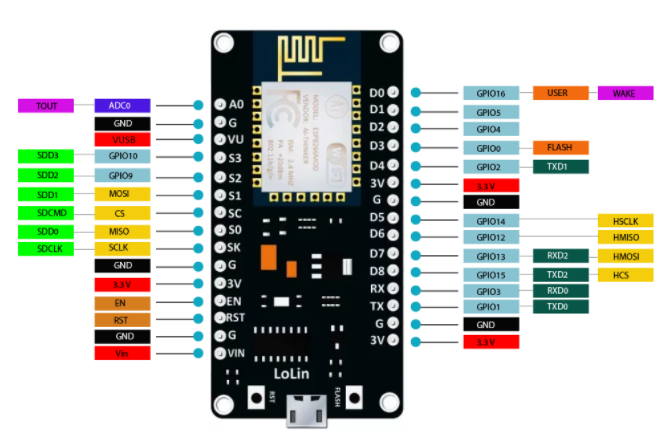
NodeMCU can be programmed using the Arduino IDE, which is made to be used by beginner programmers and less challenging for novices. Getting the Arduino IDE up and running with NodeMC will not take you longer than 5-10 minutes to do. You just need the Arduino IDE, a USB cable, and the NodeMCU, which comes with it, to use this kit. Here are the NodeMCU Development Board Pinout Configuration:

|  |  |  |
| --- | --- | --- |
| Pin Category | Name | Description |
| Power | Micro USB 3.3V, GND, Vin | Micro-USB: The Micro-USB NodeMCUU can be operated by plugging the device into a USB socket.  3.3V: 3.3V can be supplied to this pin to power the board  GND: Ground pins  Vin: External Power Supply |
| Control Pins | EN, RST | The pins and button to reset the microcontroller |
| Analog Pin | A0 | It is used to measure the analog voltage in the range of 0-3.3V |
| GPIO Pins | GPIO1 to GPIO16 | The microcontroller has 16 general purpose input-output pins |
| SPI Pins | SD1, CMD,SD0, CLK | NodeMCU has for pins for SPI communication |
| UART Pins | TXD0, RXD0, TXD2, RXD2 | It has 2 UART interfaces, UART0 (RXD0 & TXD0) and UART1 which is used to upload the firmware or program |
| I2C Pins |  | NodeMCU has I2C functionality support |

NodeMCU ESP8266 Specifications and Features:

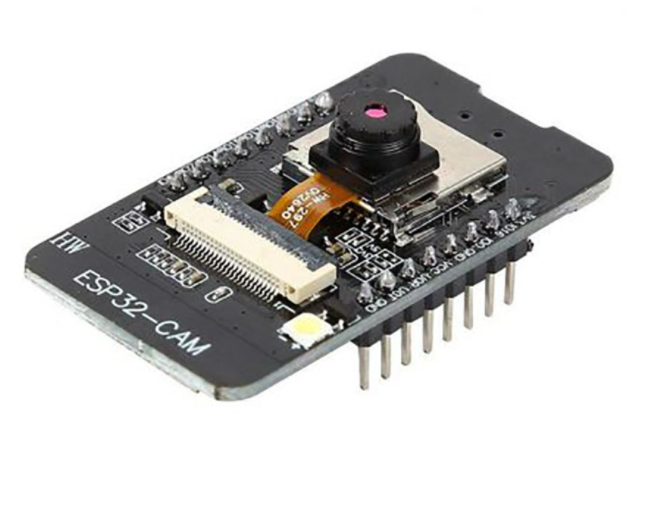
* Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
* Operating Voltage: 3.3V
* Input Voltage: 7-12V
* Digital I/O Pins (DIO): 16
* Analog Input Pins (ADC): 1
* UARTs: 1
* SPIs: 1
* I2Cs: 1
* Flash Memory: 4 MB
* SRAM: 64 KB
* Clock Speed: 80 MHz
* USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
* PCB Antenna
* Small Sized module to fit smartly inside your IoT projects

Here are the Pinout image of NodeMCU ESP8266:



### 2.2.3. ESP32 CAMERA

The ESP32 comes with integrated video and microSD socket for video, and acts as a microcontroller with an additional feature set of sensors. Camera purchases for use in general are very cheap and easy to acquire, but for cameras with advanced features like image tracking and recognition, this one is great value.



The ESP32-CAM is based upon the ESP32-S module, so it shares the same specifications.  It has the following features:

* 802.11b/g/n Wi-Fi
* Bluetooth 4.2 with BLE
* UART, SPI, I2C and PWM interfaces
* Clock speed up to 160 MHz
* Computing power up to 600 DMIPS
* 520 KB SRAM plus 4 MB PSRAM
* Supports WiFi Image Upload
* Multiple Sleep modes
* Firmware Over the Air (FOTA) upgrades possible
* 9 GPIO ports
* Built-in Flash LED

There is one minus point for this device is that the ESP32- CAM board does not have any USB port, so it cannot connect to the computer and start loading program. Instead, it needs an FTDI adapter which is the same adapter used for modifying the Sonoff 4CH Switches in the next chapter.

### 2.2.4. Sonoff 4CH Switches

<https://ewelink.coolkit.cc/?p=458>

Sonoff4-gang switches (also referred to as 4CH & 4CH R2) are built into-in 4-gang din rail enclosures. This Sonoff switches (grouped together and controlled by buttons or via the ‘EweLink’ app normally) has four-electrical appliances with one switch that is compatible with the smartphones and tablets, allowing them to be controlled separately or in turn on or all at once. Secondly, the multi gang wiring feature of App allows users to see the status of several devices on the screen simultaneously. Combining 4 Sonoff switches in a single ownership means that you will have the ability to have a single Sonoff WiFi-R2 unit.

Din rail support is used to hold the board inside the enclosure! Many circuit breakers and industrial control equipment installations use DIN rails, which are a typical D-rail system. Usually, these items are made of cold-rolled carbon steel with a chromium or zinc-plated steel.

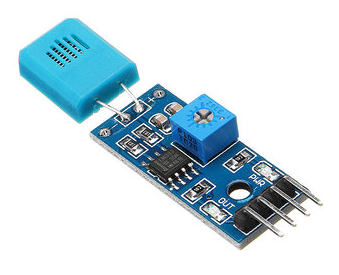
The Sonoff 4CH Wi-Fi light switch is without a doubt a member of the family, it performs all the essential functions, users can switch on or off from anywhere at any moment, with the option to set to make schedules that run once or continuously, users can all-rearrange schedules to make things go on/off at the same time, or all together to set them to stay on a given time and to single/off together.



The Sonoff 4CH in this thesis is softly modified by the aim of usages to control all the devices such as light, TV and so on using MQTT communication technology. It could be reprogramed easily by using the FTDI adapter to expand its abilities.

### 2.2.5. HR202 Sensor

This sensor uses operating voltage in range of 3.3V- 5V and has the AO as analog out put and DO as digital output. It is deal for custom humidity sensing applications, like meteorology, textile facility control, and storage building humidifying applications are those which require sensing the current humidity levels in the environment as well as the relative humidity. A current-dependent or voltage-dependent output is often called an analog or the output depending on the Comparator integrated circuit (LM393), is possible both as well as a digital. This sensor is based on HR202, having a greater linearity, wider measurement range, and lower power consumption than the hygrometric sensors are utilized in many previous designs. There is an LED indicator to show the amount of power input, and a digital indicator to show output. It is inexpensive and very low power consumption.



### 2.2.6. HC- SR505 Sensor

HC- SR505 sensor is a smaller version of the PIR (Passive Infra-Red) motion sensor which accounts for that as the small room size and reduced ability to discern subtle changes in an infant in its presence. IR technology is used to expand the beam width and the intensity. This feature is highly efficient since it is able to maintain and/increase intensity when the system control is unattended. Because of its minimum size and low-power operation, it is used in battery-powered devices as well as in others.



HC- SR505 is a three pins device which are:

Ground (GND)

Signal Output (OUT)

Positive Power Supply (+)

This sensor has wide range of operating voltage with minimum size and repeatable trigger. The module is used mainly in applications where the motion of a body emitting infaring radiations can be detected. The HC-SR505 is favored in battery systems as the unit is designed to be operated with a very low power consumption. The small size allows for the installation of the module in applications with consideration of device size. Because of its wide range detecting motion, it could be used as motion alarm, automatic lighting system, theft protection, … In this study, it will be acted as motion alarm to detect if any object is moving.

### 2.2.7. Diffused RGB LED



As what it is called, diffused 10mm LED with 3 different separated red, green and blue LED chips inside. It has 4 pins: GND and RGB pins. The purpose of this led in this project is to simulate real life RGB light but smaller version.

### 2.2.8. Google Home Mini

The Google Home Mini was invented by Google, which is a voice-activated speaker that can play music, monitor smart home devices, answer trivia questions, add items to a shopping list, schedule appointments, and play video on a Chromecast-enabled screen. Google recently introduced the ability to make phone calls from the Home speakers, as well as use them to track down your phone if you've lost it. Compact, convenient, the Home Mini smart speaker offers much of the same smart speaker features as the Echo Dot for less than the full-price at a smaller size. always-listening Wi-fi-connected speaker is something you can fit in your home that is very compact, inexpensive and gives you a broad coverage.

Google Home Mini has the size of a “donut” with many good- looking colors such as light gray, dark gray and coral red. It could be turned on by using the “OK Google” or “Hey Google” wake words. There are many activities that could be done with Google Home, but in this thesis, it will be the device which listen to the command and control things using Google API connected to openHAB2.

## 2.3. Home automation system components

### 2.3.1. openHAB

Open Home Automation Bus (openHAB) is an open source project dedicated to creating a community-oriented and pluggable platform that helps you assemble IoT devices regardless of their type, whether they use a proprietary or open protocol.

Furthermore, the features of many mobile apps have been restricted in their use and deployment because of the lack of integration and application possibilities are inherent. Unlike with Android, however, the openHAB is centric, and centers on what is wanted by the end user, which is feasible and offers a variety of ways to accomplish these tasks.

Using so-called objects, openHAB abstracts away physical devices or virtual data sources. These things free the user from having to deal with different IP addresses, MAC addresses, or the like, allowing them to concentrate on what they represent, which is normally in a human-readable and understandable format. This makes it easy to replace devices in an IoT network.

When it comes to replaceability, it's important to remember that openHAB is built on a modular architecture, which allows for the safe and inexpensive addition of new features like bindings, which provide access to other technologies, extensions, automation rules, and so on. Because of these advantages, openHAB is extremely common among home automation enthusiasts who are eager to contribute to the open source project.

OpenHAB can run on a number of operating systems, including Windows, Mac OS X, and Linux. OpenHAB is also available as pre-configured, easy-to-install images for Raspberry Pi (openHABian), Docker, PINE64, and other platforms. This allows for a relatively quick and efficient installation of the openHAB platform, which is particularly useful when only preliminary testing is needed.

The openHAB program, on the other hand, is not cloud-ready on its own. There are a few options, including hosting the openHAB runtime in the cloud, but this would necessitate a complicated networking configuration to ensure that the gateways and their connected devices can communicate with the runtime, which requires that they be on the same logical network. Fortunately, the openHAB project has a cloud companion feature that allows you to link the openHAB runtime to the cloud and expose all of the platform's related features over the Internet.

Nowaday, openHAB is proved to be a good choice for managing the IoT platform for the project due to its ease of setup and the associated advantage of focusing development efforts on delivering better user experiences in less time.

### 2.3.2. myopenHAB

myopenHAB is a cloud-based service instance that connects a running openHAB system to the Internet through the Cloud Connector extension and provides functionality like remote control, push alerts, and integration with mobile and third-party apps like Amazon Alexa, IFTTT, Google Assistant, and others.

While the openHAB Foundation gives sufficient instances of the myopenHAB cloud service instance, the cloud service instance, like everything else in the openHAB project, is open source, which means that there are builds and images that can be deployed to any cloud service provider without many issues, usually resulting in better performing instances than those hosted by the openHAB Foundation.

The key disadvantage of using the myopenHAB cloud service is that, unlike the solutions offered by AWS, GCP, or Microsoft Azure, all component management, such as updates, reliability, and any load balancing, must be done manually. But luckily, its downside of it does not affect the result of this project in general.

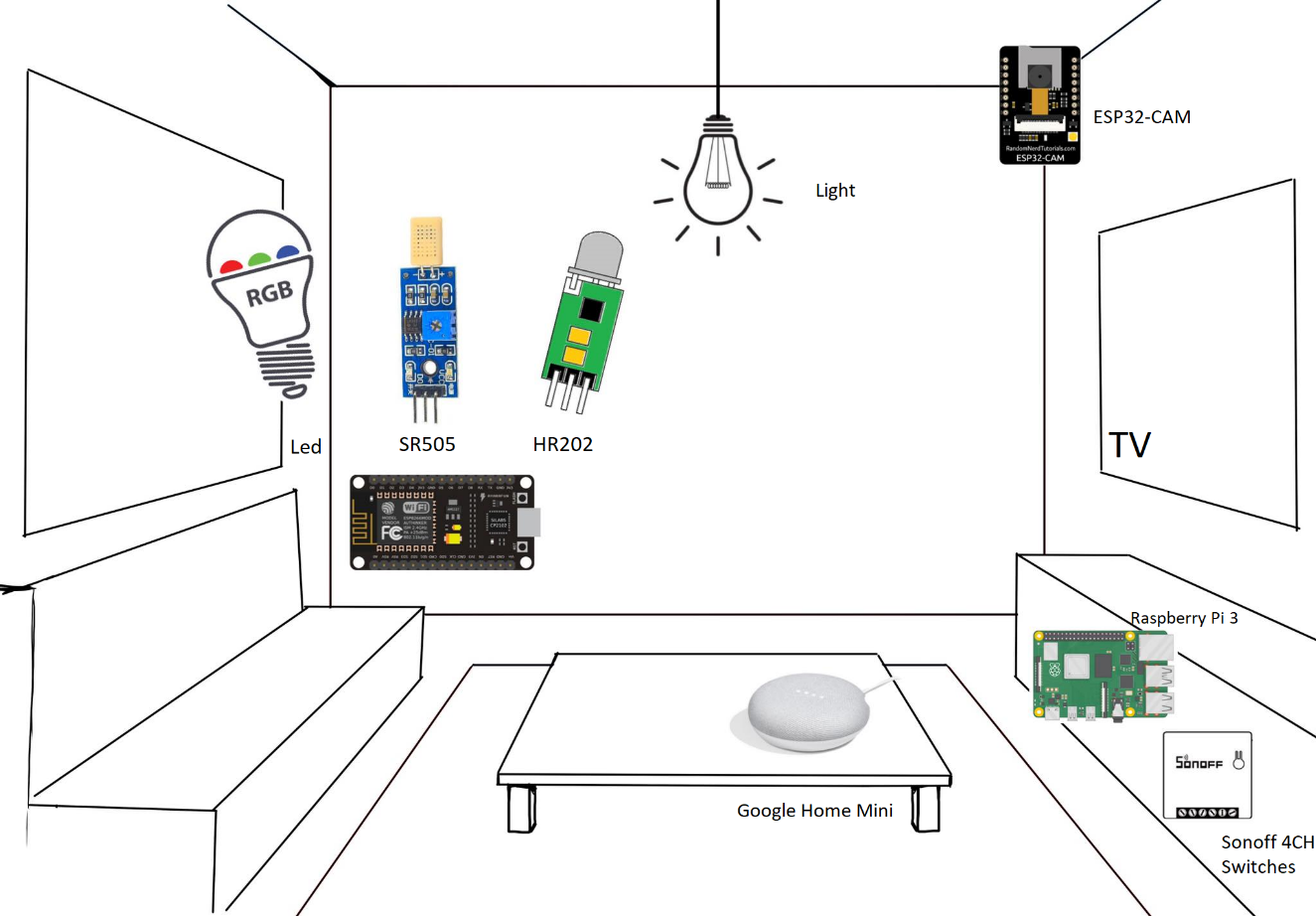
### 2.3.3. Apache2

On December 1, 1995, Apache, a web server, was launched for the first time. Under the sponsorship of the Apache Software Foundation, it is actively operated by open-community developers. Among other web servers, it is the most commonly used and popular today. Apache is a cross-platform, open-source web server that is widely used in Linux distributions. According to the Web Technology Survey, Apache is the most popular web server today, hosting 38.4% of all live websites. The fact that apache has a module-based framework makes it extremely scalable, and it has modules for authentication and URL rewriting are only a couple of the reasons why it is the most common server program.

https://www.diva-portal.org/smash/get/diva2:1439759/FULLTEXT01.pdf

## 2.4. System plan

The system plan below illustrates by the main components comprising the small house. All the sensors will be managed by the ESP8266 chip on the bread board. Google Home Mini would be used to control all the things which are connected on the same MQTT system. Raspberry Pi 3 will be the MQTT broker providing connection and communication, also hosting a web application server.



The room light and TV are controlled by Sonoff 4CH Switches together with all the sensors which are IoT-enabled, which means they can be operated from any Wi-Fi connected devices that can track their status and issue commands to them. The gateways, as well as any other devices that are responsible for the connectivity and control of these devices, are in appropriate areas within the house.

## 2.5. Decision- making process

One requirement for this project is to create a nice website application interface with a consistent look, secure and presence. It is based on JavaScript, HTML5, CSS and PHP. Moreover, it should provide abilities to support for both REST and OData JSON data models.

The table light will be chosen as the light in the real house's hallway for simulating purposes, low cost, and high practicality, while still being energy efficient and ideal for the project's scenario. Another lighting source was the RGB Diffused LED, which was chosen to demonstrate some of the power of MQTT communication.

The HR202 provides a practical humidity sensor solution for capturing the humidity of the house data due to its high reliability ratings and low cost. The SR505 sensor was selected to detect motion because of its ability to detect movement over a wide range of angles.

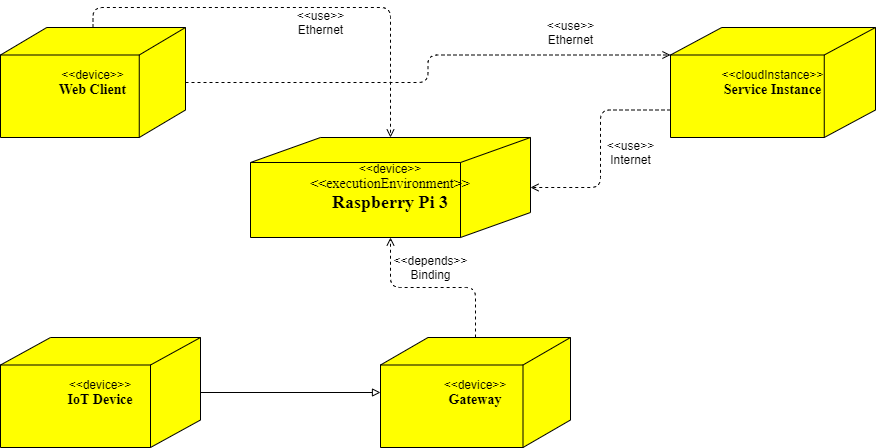
The Sonoff 4CH Switches acts as a relay and were chosen to control the other devices because of their ability to communicate via Wi-Fi. On the other hand, for the Sonoff switches, a MQTT service can be set up manually by customize its program that will publish their connected devices activities on unique channels and thus inform all interested parties.

Because of previous decisions to use commodity hardware and the project's tight budget, openHAB was chosen as the IoT platform on which the device will be hosted because it can operate on inexpensive hardware like the Raspberry Pi and offers a REST API. Because of the strong community behind openHAB, there is support for a wide range of consumer off-the-shelf devices that will be used in the project, making setup as simple as possible.

In addition, the myopenHAB cloud builds feature like remote control and third-party apps which can be linked to the Google Home Mini makes openHAB the most important choice for this project.

Many of the tedious configuration processes presented in the IoT Service Solutions for onboard IoT devices may show major developmental slowdowns. In addition, these solutions also have a subscription charge or usage fee, which cannot be distributed at phases if only proof of concept is presented.

## 2.6. User Documentation



Application requirements

Upon getting interested in this idea and this topic specifically, the requirements had been discussed with my supervisor, where the following key components were defined:

Establishing two-way communication: Ensuring that the backend system is functioning properly and that the front-end wen application can send commands and receive status notifications from the connected IoT devices.

Providing automation systems: It is possible to automate one or more onboarded devices in response to various types of actuations, such as a specific time of day, or a theme to adjust the state of another IoT device in the system, and so on.

Allowing for simple device onboarding: When a device is connected to the system, the setup should be minimal, and the onboarding procedures should need as little feedback from the user as possible.

Pleasant control panel configuration: The user should be able to use the control panel with ease, according to their expectations and needs. The design of the control panel should be flexible enough to accommodate as many different configurations as the user requires. Security is also a top priority.

Deployment scenario

The use of technological systems, automated procedures, and wired, remote-controlled devices in apartments and houses is referred to as a smart house. The functions' key goal is to enhance the quality of life and ease of use in the home. Other objectives include improved protection and energy efficiency due to wired, remote-controllable devices.

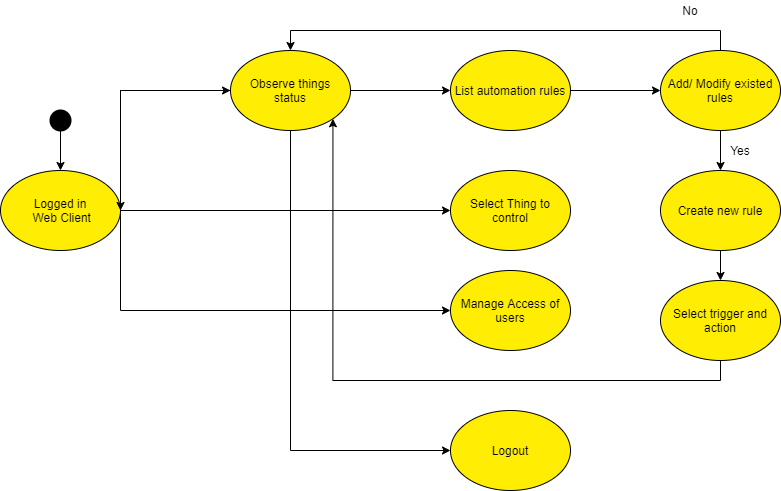
Internet of Things is awesome, and this project may be used as a base for a time-controlled feature of home appliances such as the washing machine, lighting, or coffee maker. Motion sensors, cameras, shutters, and thermostats, for example, start user-programmed processes. The central control device, with which various smart components are connected and can be managed from a PC, smartphone, or tablet, is at the heart of the smart home. For communication or system control, popular wireless protocols such as Wi-Fi or Bluetooth are could be used.

More specifically about the project, the simulation should allow prople to interact with the system by sending commands to the devices and observing their actions and monitoring the data collected from said devices. In order to allow this, a web application would be hosted from Raspberry Pi 3 as a product of HTML, CSS, JavaScript and PHP that is accessible from anywhere in the world via the Internet.

Due to the small budget, it was decided that the IoT devices such as sensors and light are enough to simulate the communication and demonstrate the real-life devices. They are all physically located in the house with easy deployment in order to give potential service efforts of Internet of Things.

Activity diagram

The activity diagrams depict the procedures for creating, reading, modifying, and removing bindings and items, as well as the dependencies between them and allowing rule-based automation. If there are no existing rules that meet our requirements, we can easily build one by specifying the theme and subsequent actions. Special requirements may be specified when designing a rule for automation, making far more flexible implementations.



# Chapter 3

# Developer Documentation

## 3.1. Installation

This section describes the mandatory steps to build the whole Internet of Things system to simulate smart house.

My aim was for the application to include two systems. The first ensures that IoT devices can be handled and that data and commands can be retrieved and released. The second system will be in charge of showing the web control panel from which the user in the family could track and control the IoT devices.

### 3.1.1. Raspberry Pi 3

#### 3.1.1.1. openHABian

Firstly, I decided to take a Raspberry Pi 3 model B for the initial phase of this project and for the proof-of-concept stage because I had it available already and the solution for deployment scenarios like these is fast and simple, especially when the costs of the system themselves are relative small. This is also a very smart device with a really low energy efficient.

Secondly, the Raspberry Pi 3 model B could easily use a ready-made image of the Raspbian Lite Linux distribution named openHABian made available by the openHAB project, which takes care of all setup phases and configuration through shell scripts included in the image. Writing the openHABian image file available on the openHAB project website to an SD card and inserting the card into the Raspberry Pi's slot are the only steps required to deploy a functioning instance of openHAB on a Raspberry Pi utilizing this process.

<https://www.smarthomeblog.net/openhab-raspberry-pi/>

Following that, I needed to connect the Raspberry Pi to the Internet, which could be done with a UDP cable or, if the Raspberry Pi has Wi-Fi capabilities, the network's service-set identifier (SSID) and pre-shared-key (PSK) should be filled out in the openhabian.conf file on the SD card before inserting it into the Raspberry Pi as values for the Wi-Fi SSID and Wi-Fi PSK respectively. The next move is to add power to the Raspberry Pi and wait 15 to 45 minutes for the openHAB instance to be mounted on the Raspberry Pi, depending on the speed of the Internet connection. The progress of the installation can be tracked using the http://openhabian or http://openhab URLs, depending on the image version, or by connecting to the Raspberry Pi's local IP address through HTTP.

Since the openHABian operating system is installed with default usernames and passwords, there is a disadvantage to this simplified version of the openHABian operating system on the Raspberry Pi. Fortunately, the openHABian Configuration Tool makes it easy to adjust these settings. The “sudo openhabian-config” command can be used to use this utility from the terminal. OpenHABian can also be mounted on other Linux distributions in the same way as every other software is. Since we'll be pulling the openHABian repository from GitHub, the only prerequisites is that we have git built on our operating system.

These commands are to manually install the openHABian:

|  |
| --- |
| sudo git clone https://github.com/openhab/openhabian.git /opt/openhabian  sudo ln -s /opt/openhabian/openhabian-setup.sh /usr/local/bin/openhabian-config  sudo openhabian-config |

#### 3.1.1.2. Apache2

Firstly, I need to make sure we have an up-to-date computer after installed the openHABian instance and before downloading the Apache2 server. To do so, we'll need administrator privileges, which we can get by using the sudo command. After the Raspberry is up to date, it is enough requirement to install Apache server now. These commands are for update and install the Apache server:

|  |
| --- |
| sudo apt update  sudo apt upgrade  sudo apt update  sudo apt install apache2 |

Following that, we will need to consider changing the server’s files permissions so that we could modify the inventory easier:

|  |
| --- |
| sudo chown -R pi:www-data /var/www/html  sudo chmod -R 770 /var/www/html/ |

The website's root directory is /var/www/html, which is used by Apache. This means that Apache searches for the file in "/var/www/html" when we call the Raspberry on port 80 (http).

The home page of Apache which we could access is the IP of the Raspberry Pi in the local network. Moreover, if users type http://”IPofRaspberryPi”/example into the browser, Apache can search for the "example" file in the "/var/www/html" directory.

After all, we can now use your Raspberry to build an internal website using HTML, CSS, and JavaScript. Those who can, however, want to make easy interactions between the site and the customer. To allow the user to log, for example. This will necessitate the use of PHP.

#### 3.1.1.3. PHP

PHP is a free programming language that is maintained by the PHP Foundation, Zend Enterprise, and a number of other organizations.

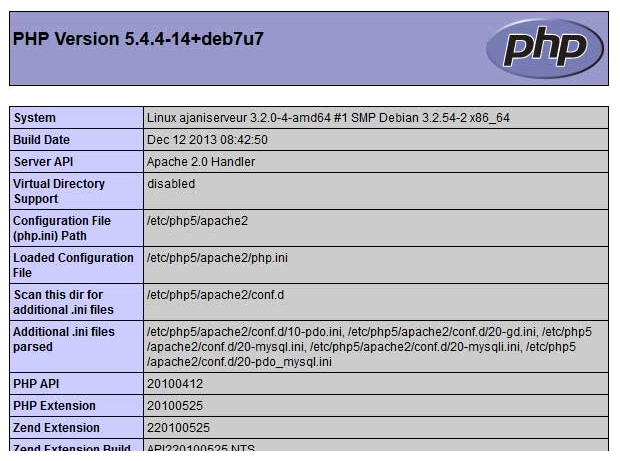
PHP is mostly used to render a website interactive, which means that the user sends data to the server, which then returns updated responses based on that data. A static site, on the other hand, does not respond to the information supplied by the customer. It will still provide the same content since it is stored as a file once. It's one of the most popular programming languages, and it's still the most popular for web development.

To install the PHP on the Raspberry Pi 3, I simply used these following commands:

|  |
| --- |
| cd /var/www/html/  sudo apt install php php-mbstring  sudo rm index.html  echo “<?php phpinfo();?>” > index.php |

Because we already had the index page of Apache2 above, we would need to delete the index file after in order to create a new php main page. After all, we could check the page with the same method used for Apache2, typing this link to the browser: <http://”IPofRaspberry>”.

The webserver home page should look like:



The Developer Documentation (or Developer’s Manual) should contain

the detailed specification of the problem,

the detailed description of the used methods, the definitions of the used no- tions,

the description of the logical and physical structure of the software (data structures, databases, modules etc.),

the testing plan and the results of the tests.

# Bibliography

[1] Somebody, something.